1:

INPUT: array size m;Lower limit of value L ;Lower limit of value R;whether the values is *unique*;

OUTPUT:array a

$a \leftarrow\ Emptyarray()$

$n \leftarrow\ R - L + 1$

if *unique* is *true* **then**

if $ln(n+1) < 1 + log((n-m)+1)$ **then**

for $i \leftarrow 1$ to m **do**

while u exits in a **do**

$u \leftarrow rand(L, R)$

end while

append Integer u to a

end for

else

for $i \leftarrow L$ to R **do**

append i to a

end for

$a \leftarrow\ shuffle(a[1, m])$

end if

else

for $i \leftarrow L$ to R **do**

$u \leftarrow rand(L, R)$

while u exits in a **do**

$u \leftarrow rand(L, R)$

end while

append Integer u to a

end for4

end if

return a

2

INPUT: Number of nodes of the tree n;

OUTPUT: Pairs Arrays *edge*

for $i \leftarrow 2$ to n **do**

$u \leftarrow rand(1, i-1)$

$v \leftarrow i$

edge append $\{u, v\}$

end for

return *edge*

3:

INPUT: Number of nodes of the graph n;Number of edge of the graph m;

OUTPUT: Pairs Arrays *edge*

for $i \leftarrow 2$ to n **do**

$u \leftarrow rand(1, i-1)$

$v \leftarrow i$

edge append $\{u, v\}$

end for

$Upper\ limit \leftarrow n * (n-1)/2$

if $ln(Upper\ limit + 1) < 1 + log((Upper\ limit - m) + 1)$ **then**

for $i \leftarrow n$ to m **do**

while $u = v$ *or edge* exits $\{u, v\}$ **do**

$u \leftarrow rand(1, n)$

$v \leftarrow rand(1, n)$

if $u > v$ **do**

$swap(u, v)$

end do

end while

edge append $\{u, v\}$

end for

else

while $size(a) < m$ **do**

a append $\{i, j\}$ where $1 \leq i < j \leq n$ and $\{i, j\}$ not exits in a

end while

end if

return a

4:

INPUT: Number of nodes of the graph n;Number of edge of the graph m;

OUTPUT: Pairs Arrays *edge*

for $i \leftarrow 2$ to n **do**

$u \leftarrow rand(1, i-1)$

$v \leftarrow i$

edge append $\{u, v\}$

end for

$Upper\ limit \leftarrow n * (n-1)$

if $ln(Upper\ limit + 1) < 1 + log((Upper\ limit - m) + 1)$ **then**

$no\ ring \leftarrow true$

for $i \leftarrow n$ to m **do**

while $u = v$ *or edge* exits $\{u, v\}$ do

$u \leftarrow rand(1, n)$

$v \leftarrow rand(1, n)$

if $no\ ring = true$ and $u < v$ **do**

$swap(u, v)$

end if

end while

edge append $\{u, v\}$

$no\ ring \leftarrow false$

end for

else

a append $\{j, i\}$ where $1 \leq i < j \leq n$ and $\{j, i\}$ not exits in a

while $size(a) < m$ **do**

a append $\{i, j\}$ where $\{i, j\}$ not exits in a

end while

end if

return a

5:

INPUT: Number of nodes of the graph n;Number of edge of the graph m;struct Arrays *edge with value*;

OUTPUT:*ture* or $false$

$graph[][] \leftarrow$ empty Two-dimensional arrays

for $\{u, v, w\}$ in *edge with value* **do**

$graph[u][v] = w$

end for

$stk \leftarrow$ empty *stack*

$st[] \leftarrow \{true\}$

$dist[] \leftarrow \{0\}$

$cnt[] \leftarrow \{0\}$

stk push i where $1 \leq i \leq n$

while $size(stk) \neq 0$ **do**

$t \leftarrow$the top of stk

pop the top of stk

$st[t] = false$

for $target \leftarrow 1$ to n **do**

if $graph[t][target]$ no null and $dist[target] > dist[t] + graph[t][target]$ **do**

$dist[target] \leftarrow dist[t] + graph[t][target]$

$cnt[target] \leftarrow cnt[t] + 1$

if $cnt[target] \geq n$ **do**

return *true*

end if

if $st[target] = false$ **do**

$st[target] = true$

push *target* into stk

end if

end if

end for

end while

return $false$

6:

INPUT: Number of nodes of the graph n;Number of edge of the graph m;

OUTPUT:Pairs Arrays $edge$

$graph[][]$ ←empty array

for $i \leftarrow 2$ to n **do**

$u \leftarrow rand(1, i-1)$

$v \leftarrow i$

$edge$ append $\{u, v\}$

$graph[u][v] = true$

end for

$set[2]$ ←empty array

function $dfs(u, father, color)$

append u to $set[color]$

for $j \leftarrow 1$ to n **do**

if $graph[u][j] = true$ and $j \neq father$ do

call $dfs(j, u, color\ xor\ 1)$

end if

end for

end funtion

call $dfs(rand(1, n), 0, 0)$

for $i \leftarrow n$ to m **do**

while $u \neq v$ and $\{i, j\}$ no exits in $edge$ **do**

$u = rand(set[0])$

$v = rand(set[1])$

end while

append $\{u, v\}$ to $edge$

end for

return $edge$

7:

INPUT: Number of nodes of the graph n;

OUTPUT:Pairs Arrays $edge$

$top[] = \{0\}$

$fa[] = \{0\}$

$graph[][] = 0$

for $i \leftarrow 2$ to n **do**

$u \leftarrow rand(1, i-1)$

$v \leftarrow i$

$edge$ append $\{u, v\}$

$graph[u][v] = true$

end for

function $dfs(x, father, grand\ father)$

$top[x] = grand\ father$

$fa[x] = father$

$link\ son \leftarrow j$ where j is any one of $graph[x][j] = true$

call $dfs(link\ son, x, grand\ father)$

for $j \leftarrow 1$ to n **do**

if $graph[x][j] = true$ and $j \neq link\ son$ **do**

call $dfs(j, x, j)$

end if

end for

end function

$root \leftarrow rand(1, n)$

call $dfs(root, 0, root)$

$vis[] = \{false\}$

for $i \leftarrow 1$ to n **do**

if $vis[top[i]] = false$ and $top[i] \neq i$ and $top[i] \neq fa[i]$ **do**

$vis[top[i]] = true$

append $\{i, top[i]\}$ to $edge$

end if

end for

return edge